

มหาวิทยาลัยสงขลานครินทร์

คณะวิศวกรรมศาสตร์



สอบปลายภาค: ภาคการศึกษาที่ 2

ปีการศึกษา: 2552

วันที่สอบ: 23 กุมภาพันธ์ 2553

เวลาสอบ: 9.00-12.00

รหัสวิชา: 240-208

ห้องสอบ:

ชื่อวิชา: FUNDAMENTALS OF COMPUTER ARCHITECTURE

คำสั่ง

อ่านรายละเอียดของข้อสอบและคำแนะนำให้เข้าใจก่อนเริ่มทำข้อสอบ

อนุญาต : เครื่องเขียนต่างๆ ปากกา หรือดินสอ

: กระดาษโน้ตซึ่งเขียนด้วยลายมือขนาด A4 จำนวน 1 แผ่น (ให้เขียนด้วยลายมือเท่านั้น ห้ามพิมพ์ ห้ามมีแผ่นปะติด ห้ามถ่ายเอกสาร)

: เครื่องคิดเลข

ไม่อนุญาต : Computer Notebook, Pocket PC, หนังสือ และสมุดต่างๆ

เวลา : 3 ชั่วโมง (180 นาที)

คำแนะนำ

- ข้อสอบ 14 หน้า(รวมทั้งใบปะหน้า) หน้า 12-14 เป็นข้อมูลสำหรับใช้ตอบคำถาม
- โจทย์มีจำนวน 8 ข้อ ให้ทำทุกข้อ และเขียนคำตอบลงในข้อสอบเท่านั้น
- อนุญาตให้ใช้ดินสอในการทำข้อสอบได้ กรณีเขียนไม่ชัดหรืออ่านไม่ออก จะถือว่าคำตอบนั้นผิด
- อ่านคำสั่งในแต่ละข้อให้เข้าใจก่อนลงมือทำ
- ให้เขียนชื่อ-นามสกุลและรหัสนักศึกษาในข้อสอบทุกแผ่น แผ่นใดไม่เขียนหรือเขียนไม่ครบจะถูกตัดคะแนนแผ่นละ 1 คะแนน
- อนุญาตให้ทศเลขลงด้านหลังของข้อสอบได้

-ทุกริตีในการสอบมีโทษขั้นต่ำปรับตกในรายวิชานี้และพักการเรียน 1 ภาคการศึกษา-

1. จงอธิบายข้อแตกต่างของสถาปัตยกรรม RISC, CISC และ VLIW มาอย่างละเอียด (3 คะแนน)

คุณสมบัติ	RISC	CISC	VLIW
ลักษณะคำสั่ง			
ความยาวคำสั่ง			
วงจร Control unit ที่ใช้			
จำนวนคำสั่ง			
การกินพลังงาน			

2. จงอธิบายการประมวลผลคำสั่งต่อไปนี้ (3 คะแนน)

SIMD.....

.....

.....

.....

SISD.....

.....

.....

.....

MIMD.....

.....

.....

.....

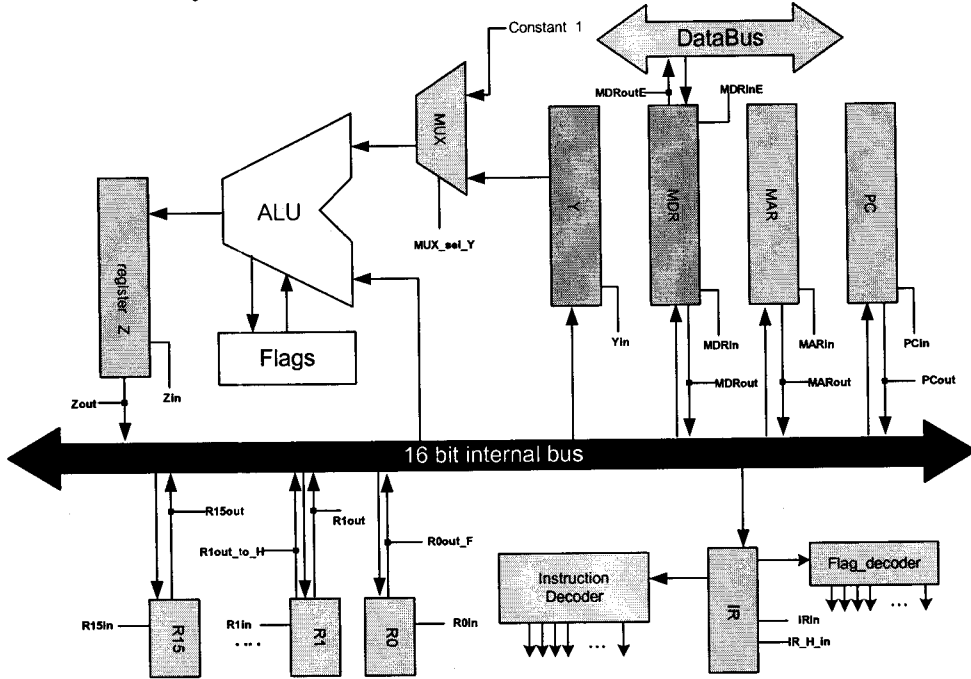
.....

7 จงขีดเครื่องหมาย X เพื่อเลือกคำตอบที่ถูกต้องมากที่สุดในแต่ละข้อเพียงคำตอบเดียว (10 คะแนน)
 คำแนะนำ ช่องคำตอบใดที่เป็นสีดำ หมายถึงนักศึกษาไม่สามารถเลือกคำตอบช่องนั้นได้

คุณสมบัตินี้คือไปนี้เป็นคุณสมบัตินี้ของสถาปัตยกรรมคอมพิวเตอร์แบบใด

คุณสมบัตินี้	CISC	RISC	VLIW	Von-Neuman	Harvard
1. มีคำสั่งให้ใช้งานจำนวนมาก					
2. คำสั่งมีเฉพาะคำสั่งพื้นฐาน					
3. ชุดคำสั่งมีความซับซ้อนในการทำงานสูง					
4. มีการกำหนดการทำงานแบบขนานไว้อย่างชัดเจนในตัวชุดคำสั่ง					
5. มี Execution Unit หลายๆ ตัว					
6. แยกหน่วยความจำโปรแกรมและหน่วยความจำข้อมูลออกจากกัน โดยเด็ดขาด					
7. โครงสร้างมีความซับซ้อนสูง					
8. มี addressing mode ให้เลือกใช้งานเป็นจำนวนมาก					
9. มีความยาวของรหัสคำสั่งไม่คงที่					
10. โปรแกรมที่คอมไพล์แล้วมักมีขนาดใหญ่เมื่อเทียบกับสถาปัตยกรรมอื่น					
11. มักจะใช้ในเครื่องคอมพิวเตอร์ PC โดยทั่วไป					
12. มักใช้ในงานประเภท Digital Signal Processing (DSP)					
13. หน่วยความจำตำแหน่งหนึ่งๆ สามารถเก็บได้ทั้งโปรแกรมและข้อมูล					
14. ต้องการความสามารถของคอมไพเลอร์ในการจัดการให้เป็นแบบขนาน					
15. เฉพาะคำสั่ง Load/Store เท่านั้นที่สามารถเข้าถึงหน่วยความจำได้					
16. มีจำนวน Register เป็นจำนวนน้อย					
17. มักใช้ Control Unit แบบ Hardwired					
18. ซีพียู Atom					
19. ซีพียู AMD Duron					
20. ซีพียู ตระกูล MCS-51					
21. ซีพียู ตระกูล IA64					
22. ซีพียู ARM7TDMI					
23. ซีพียู Itanium					
24. ซีพียู Power-PC-G4					

8. กำหนดให้ซีพียู Simple1 มีคำสั่งใช้งานจำนวน 16 คำสั่ง และมี datapath ดังรูป



หากซีพียูมีวงจร control unit แบบ hardwire แล้ว จงออกแบบวงจรสร้างสัญญาณควบคุมของซีพียูจำนวน 3 สัญญาณ คือ MDRoutE, ALUadd และ PCin

คำแนะนำ ให้ดูภาคผนวก 1-3 ซึ่งอยู่ท้ายข้อสอบประกอบ

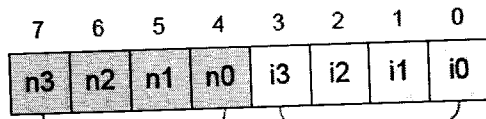
วงจรสร้างสัญญาณ MDRoutE (3 คะแนน)

วงจรสร้างสัญญาณ ALU_add (6 คะแนน)

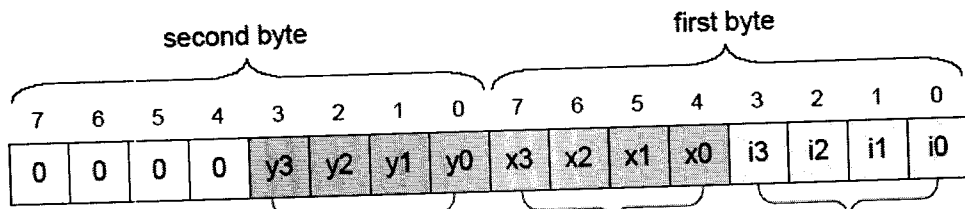
ภาคผนวก 1- ชุดคำสั่งของ Simple1

คำสั่ง	ความหมาย	ความยาวของคำสั่ง	ผลกระทบต่อแฟลก				
			C	S	Z	O	P
			F	F	F	V	F
INV Rn	กลับค่าบิตของรีจิสเตอร์ Rn เป็นตรงกันข้าม	1 byte			x		x
SETF b,v	เซ็ทค่าบิตของแฟลกที่ระบุให้มีค่าตามที่กำหนด เช่น SETF CF,1	1 byte	?	?	?	?	?
Load Rn,[R1R0]	ก๊อปปี้ค่าในหน่วยความจำตำแหน่งที่ระบุมาใส่ในรีจิสเตอร์ Rn	1 byte					
Store [R1R0], Rn	ก๊อปปี้ค่าในรีจิสเตอร์ Rn มาใส่ในหน่วยความจำตำแหน่งที่ระบุ	1 byte					
Branch [R1R0]	บรานซ์แบบไม่มีเงื่อนไขไปยังแอดเดรสที่ระบุ	1 byte					
Branch ZF, [R1R0]	บรานซ์ไปยังแอดเดรสที่ระบุหาก ZF=1	1 byte					
Branch SF, [R1R0]	บรานซ์ไปยังแอดเดรสที่ระบุหาก SF=1	1 byte					
Branch PF, [R1R0]	บรานซ์ไปยังแอดเดรสที่ระบุหาก PF=1	1 byte					
Branch OV, [R1R0]	บรานซ์ไปยังแอดเดรสที่ระบุหาก OV=1	1 byte					
Branch CF, [R1R0]	บรานซ์ไปยังแอดเดรสที่ระบุหาก CF=1	1 byte					
ADD Rx,Ry	$Rx \leq Rx + Ry$ (Addition)	2 bytes	x	x	x	x	x
ADDC Rx,Ry	$Rx \leq Rx + Ry + CF$ (Addition with carry flag)	2 bytes	x	x	x	x	x
SUBB Rx,Ry	$Rx \leq Rx - Ry - CF$ (Subtraction with borrow flag)	2 bytes	x	x	x	x	x
AND Rx,Ry	$Rx \leq Rx \& Ry$	2 bytes			x		x
OR Rx,Ry	$Rx \leq Rx Ry$	2 bytes			x		x
Load Rx,#data8	อ่านค่าคงที่ค่า #data8 มาใส่ในรีจิสเตอร์ Rx	2 bytes					

ภาคผนวก 2-ฟอร์มแมตของคำสั่งของ Simple1



- | | |
|----------|------------------------|
| Rn | |
| 0000=R0 | 0000 = Branch |
| 0001=R1 | 0001 = INV |
| 0010=R2 | 0010 = SETF |
| ... | 0011 = LOAD Rn,[R1,R0] |
| ... | 0100 = STORE |
| ... | 0101 = Branch PF |
| ... | 0110 = Branch SF |
| 1111=R15 | 0111 = Branch CF |
| | 1000 = Branch OV |
| | 1001 = Branch ZF |



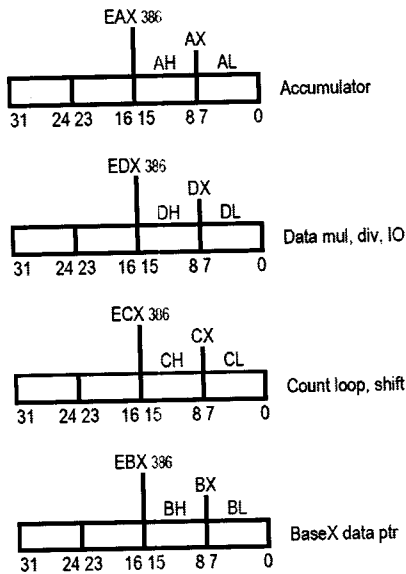
- | | | |
|----------|----------|-----------------------|
| Ry | Rx | |
| 0000=R0 | 0000=R0 | 1010 = ADD |
| 0001=R1 | 0001=R1 | 1011 = ADDC |
| 0010=R2 | 0010=R2 | 1100 = SUBB |
| ... | ... | 1101 = AND |
| ... | ... | 1110 = OR |
| 1111=R15 | 1111=R15 | 1111 = LOAD Rx,#data8 |

ภาคผนวก 3- 80x86 Instruction Set table

JUMPS (flags remain unchanged)				Name	Comment	Code	Operation
CALL	Call subroutine	CALL Proc		RET	Return from subroutine	RET	
JMP	Jump	JMP Dest					
JE	Jump if Equal	JE Dest	(= JZ)	JNE	Jump if not Equal	JNE Dest	(= JNZ)
JZ	Jump if Zero	JZ Dest	(= JE)	JNZ	Jump if not Zero	JNZ Dest	(= JNE)
JCXZ	Jump if CX Zero	JCXZ Dest		JECXZ	Jump if ECX Zero	JECXZ Dest	386
JP	Jump if Parity (Parity Even)	JP Dest	(= JPE)	JNP	Jump if no Parity (Parity Odd)	JNP Dest	(= JPO)
JPE	Jump if Parity Even	JPE Dest	(= JP)	JPO	Jump if Parity Odd	JPO Dest	(= JNP)

JUMPS Unsigned (Cardinal)				JUMPS Signed (Integer)			
JA	Jump if Above	JA Dest	(= JNBE)	JG	Jump if Greater	JG Dest	(= JNLE)
JAE	Jump if Above or Equal	JAE Dest	(= JNB = JNC)	JGE	Jump if Greater or Equal	JGE Dest	(= JNL)
JB	Jump if Below	JB Dest	(= JNAE = JC)	JL	Jump if Less	JL Dest	(= JNGE)
JBE	Jump if Below or Equal	JBE Dest	(= JNA)	JLE	Jump if Less or Equal	JLE Dest	(= JNG)
JNA	Jump if not Above	JNA Dest	(= JBE)	JNG	Jump if not Greater	JNG Dest	(= JLE)
JNAE	Jump if not Above or Equal	JNAE Dest	(= JB = JC)	JNGE	Jump if not Greater or Equal	JNGE Dest	(= JL)
JNB	Jump if not Below	JNB Dest	(= JAE = JNC)	JNL	Jump if not Less	JNL Dest	(= JGE)
JNBE	Jump if not Below or Equal	JNBE Dest	(= JA)	JNLE	Jump if not Less or Equal	JNLE Dest	(= JG)
JC	Jump if Carry	JC Dest		JO	Jump if Overflow	JO Dest	
JNC	Jump if no Carry	JNC Dest		JNO	Jump if no Overflow	JNO Dest	
				JS	Jump if Sign (= negative)	JS Dest	
				JNS	Jump if no Sign (= positive)	JNS Dest	

General Registers:



Example:

```

.DOSSEG ; Demo program
.MODEL SMALL
.STACK 1024

Two EQU 2 ; Const
.DATA
VarB DB ? ; define Byte, any value
VarW DW 1010b ; define Word, binary
VarW2 DW 257 ; define Word, decimal
VarD DD 0AFFFFh ; define Doubleword, hex
S DB "Hello!",0 ; define String
.CODE
main: MOV AX,DGROUP ; resolved by linker
MOV DS,AX ; init datasegment reg
MOV [VarB],42 ; init VarB
MOV [VarD],-7 ; set VarD
MOV BX,Offset[S] ; addr of "H" of "Hello !"
MOV AX,[VarW] ; get value into accumulator
ADD AX,[VarW2] ; add VarW2 to AX
MOV [VarW2],AX ; store AX in VarW2
MOV AX,4C00h ; back to system
INT 21h
END main
    
```

Flags:

Control Flags (how instructions are carried out):
D: Direction 1 = string op's process down from high to low address
I: Interrupt whether interrupts can occur. 1 = enabled
T: Trap single step for debugging

Status Flags (result of operations):

C: Carry result of unsigned op. is too large or below zero. 1 = carry/borrow
O: Overflow result of signed op. is too large or small. 1 = overflow/underflow
S: Sign sign of result. Reasonable for Integer only. 1 = neg. / 0 = pos.
Z: Zero result of operation is zero. 1 = zero
A: Aux. carry similar to Carry but restricted to the low nibble only
P: Parity 1 = result has even number of set bits